



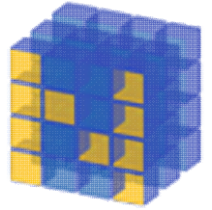
Distance Learning System

Upotreba biblioteka NumPy i matplotlib

Python Data Access

vladimir.maric@link.co.rs

NumPy



- NumPy je Python biblioteka za napredno rukovanje nizovima / matricama
- Neizostavni je sastojak paketa za rad sa podacima

pip install numpy

```
import numpy as np
```

Kreiranje numpy niza

- NumPy najvećim delom radi sa objektima tipa ndarray
- Ndarray je višedimenzionalni niz
- Generisanje ndarray objekta, vrši se pozivom funkcije array
- Funkciji array prosleđuje se nabrojivi tip ili se niz generiše funkcijom **arange** ili transformacijom python sekvence

```
arr = np.array([1,2,3,4])
```

```
arr = np.arange(1,5)
```

- Dobijeni objekat je tipiziran i tretiran kao matrica
- Svaki niz (red) matrice, naziva se osa (**axis**)

```
[1 2 3 4]
```

Tipizacija numpy niza

- Numpy nizovi su strogo tipizirani
- Tip niza se određuje automatski

```
arr = np.array([1,2,3,4])
```

Tip je **int64**

```
arr = np.array([1.,2,3,4])
```

Tip je **float64**

```
arr = np.array([0,1,2,3],  
               dtype=np.bool)
```

Tip je **bool**

Dostupni tipovi podataka

<https://numpy.org/devdocs/user/basics.types.html>

Numpy type	Description
np.int8	Byte (-128 to 127)
np.int16	Integer (-32768 to 32767)
np.int32	Integer (-2147483648 to 2147483647)
np.int64	Integer (-9223372036854775808 to 9223372036854775807)
np.uint8	Unsigned integer (0 to 255)
np.uint16	Unsigned integer (0 to 65535)
np.uint32	Unsigned integer (0 to 4294967295)
np.uint64	Unsigned integer (0 to 18446744073709551615)
np.intp	Integer used for indexing, typically the same as ssize_t
np.uintp	Integer large enough to hold a pointer
np.float32	Note that this matches the precision of the builtin python float
np.float64 / np.float_	
np.complex64	Complex number, represented by two 32-bit floats (real and imaginary components)
np.complex128 / np.complex_	

Višedimenzionalni nizovi

- Numpy je u stanju da detektuje višedimenzionalne nizove, i podržava različite aritmetičke operacije nad njima

```
arr = np.array([[1,2,3],[4,5,6]])  
print(arr)
```



```
[[1 2 3]  
 [4 5 6]]
```

- Numpy je u stanju da detektuje višedimenzionalne nizove, i podržava različite aritmetičke operacije nad njima

```
print(arr*2)
```

```
[[ 2  4  6]  
 [ 8 10 12]]
```

```
print(arr+2)
```

```
[[3 4 5]  
 [6 7 8]]
```

```
print( np.array([1,2,3])+  
       np.array([2,3,4]))
```

```
[3 5 7]
```

Aritmetika nad nizovima

- Numpy primenjuje aritmetičke operacije nad nizovima ukoliko se dimenzije mogu poklopiti:

```
a = np.array([[1,2],[3,4]])  
b = np.array([[2,3],[4,5]])  
print(a*b)
```

1*2 , 2*3
3*4 , 4*5

```
[[ 2  6]  
 [12 20]]
```

Aritmetika nad nizovima

- Ukoliko se dimenzije ne poklapaju, numpy prijavljuje grešku

```
a = np.array([1,2,3])  
b = np.array([1,2])  
print(a*b)
```

1*1 , 2*2 , 3*?

```
ValueError: operands could not be broadcast together with shapes (3,) (2,)
```


Preoblikovanje nizova

- Numpy može preoblikovati višedimenzionalne nizove i tada se takođe mora voditi računa o tome da ciljni oblik odgovara trenutnom sadržaju

Može

```
arr = np.array([[1,2,3,4],[5,6,7,8]])  
arr = arr.reshape(4,2)  
print(arr)
```

```
[[1 2]  
 [3 4]  
 [5 6]  
 [7 8]]
```

Ne može

```
arr = np.array([[1,2,3,4],[5,6,7,8]])  
arr = arr.reshape(3,2)  
print(arr)
```

```
ValueError: cannot reshape array of size 8 into shape (3,2)
```

Kompresovanje nizova

- Metod squeeze, svodi niz na vrednosti jednodimenzionalnih nizova

```
arr = [  
    [  
        [1], [2], [3]  
    ]  
]  
print(np.squeeze(arr))
```

[1 2 3]

Transponovanje matrice

```
arr = np.array([
    [1,2,3,4,5,6,7,8,9,10],
    [1,2,3,4,5,6,7,8,9,10]
])
print(arr.T)
```

```
print(arr.T)
```

Matrična aritmetika nad nizovima

- Definisanje jedinične matrice

```
mat = np.eye(4)
```

```
mat = np.identity(4)
```

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

- Definisanje jedinične matrice

Množenje matrica

```
a = np.array([1,2])  
b = np.array([[1,2],[3,4]])  
c = np.matmul(a,b)
```

```
a = np.array([1,2])  
b = np.array([[1,2],[3,4]])  
c = a@b  
print(c)
```


$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 7 \\ 10 \end{bmatrix}$$

Strukturirani nizovi

- Numpy omogočava simuliranje strukture podataka kroz specijalno definisane višedimenzionalne nizove

```
tp=[('name', 'U10'), ('price', float)]  
  
b = np.array([("Phone",125.99),("Bicycle",100.5),("TV",82.22)],tp)  
  
print(  
    b[0]["name"],  
    b[0]["price"]  
)
```

Matplotlib biblioteka



- Matplotlib je biblioteka za crtanje
- Matplotlib se najčešće koristi za crtanje grafikona na osnovu kolekcija podataka

pip install matplotlib

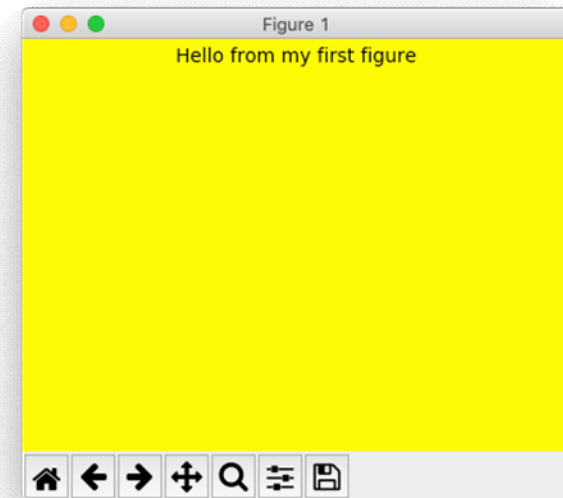
Figure klasa

- Učitavanje modula

```
import matplotlib.pyplot as plt
```

- U osnovi svakog prozora je objekat klase Figure

```
p = plt.figure(  
    facecolor="yellow"  
)  
p.suptitle(  
    "Hello from my first figure",  
    fontsize="medium"  
)  
plt.show()
```



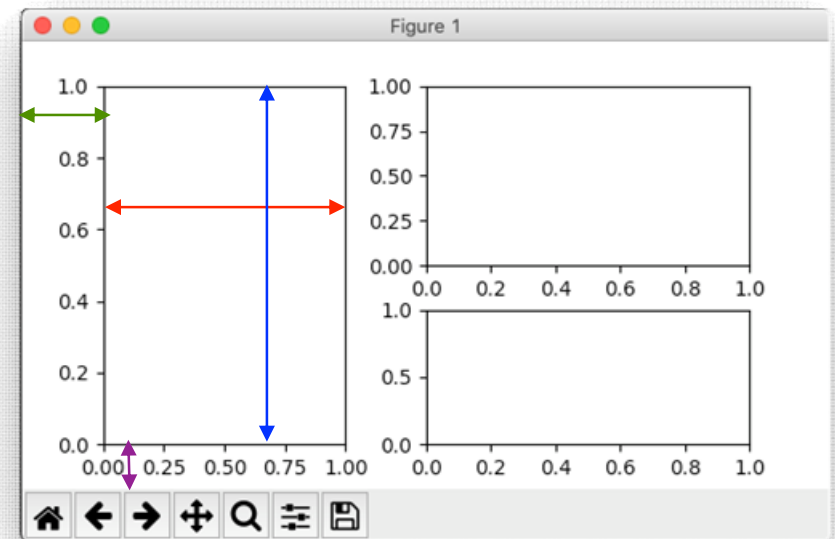
Ose

- Svaka figura sastoji se od osa (Axes)
- Ose su individualni grafikoni na kojima je moguće izvršiti iscrtavanje

```
f = plt.figure()
f.add_axes([0.1, 0.1, 0.3, 0.8])
f.add_axes([0.5, 0.5, 0.4, 0.4])
f.add_axes([0.5, 0.1, 0.4, 0.3])
plt.show()
```

Diagram illustrating the parameters for `add_axes()` using a 2x2 grid of axes:

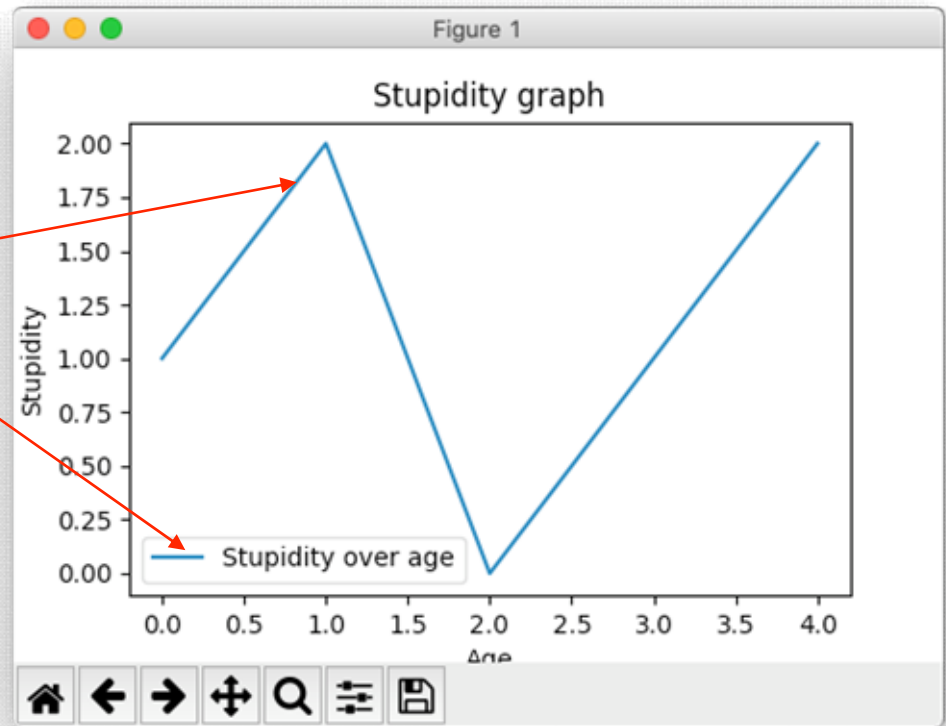
- x**: x-axis position (0.1)
- y**: y-axis position (0.1)
- w**: width (0.3)
- h**: height (0.8)



Uvođenje elemenata

- Metod plot automatski generiše figuru i on je dovoljan za jedan grafikon

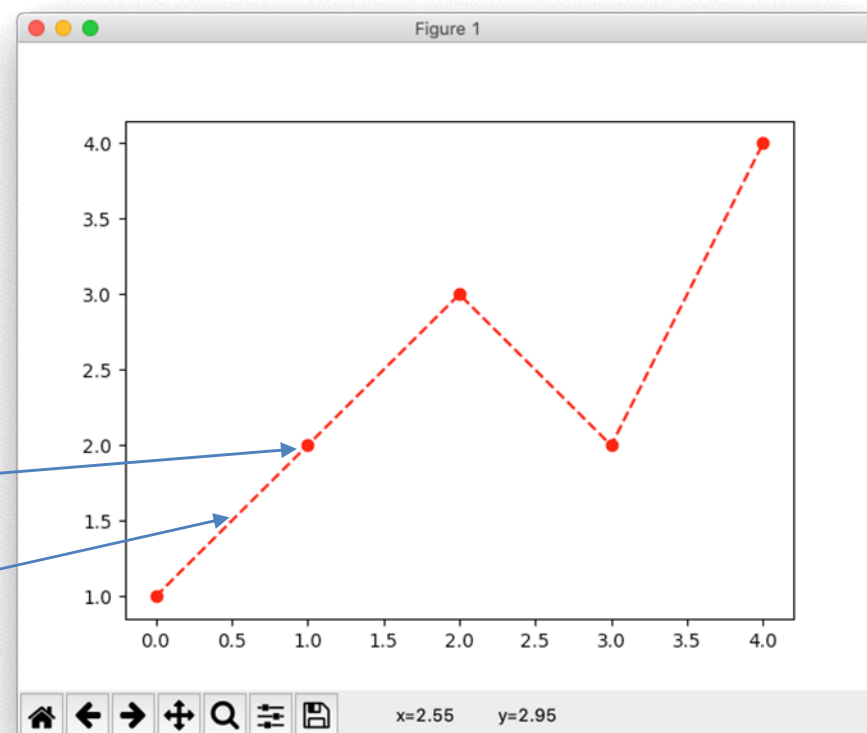
```
plt.plot(  
    [1,2,0,1,2],  
    label='Stupidity over age')  
plt.title("Stupidity graph")  
plt.legend()  
plt.ylabel("Stupidity")  
plt.xlabel("Age")  
plt.show()
```



Crtanje oblika (linija)

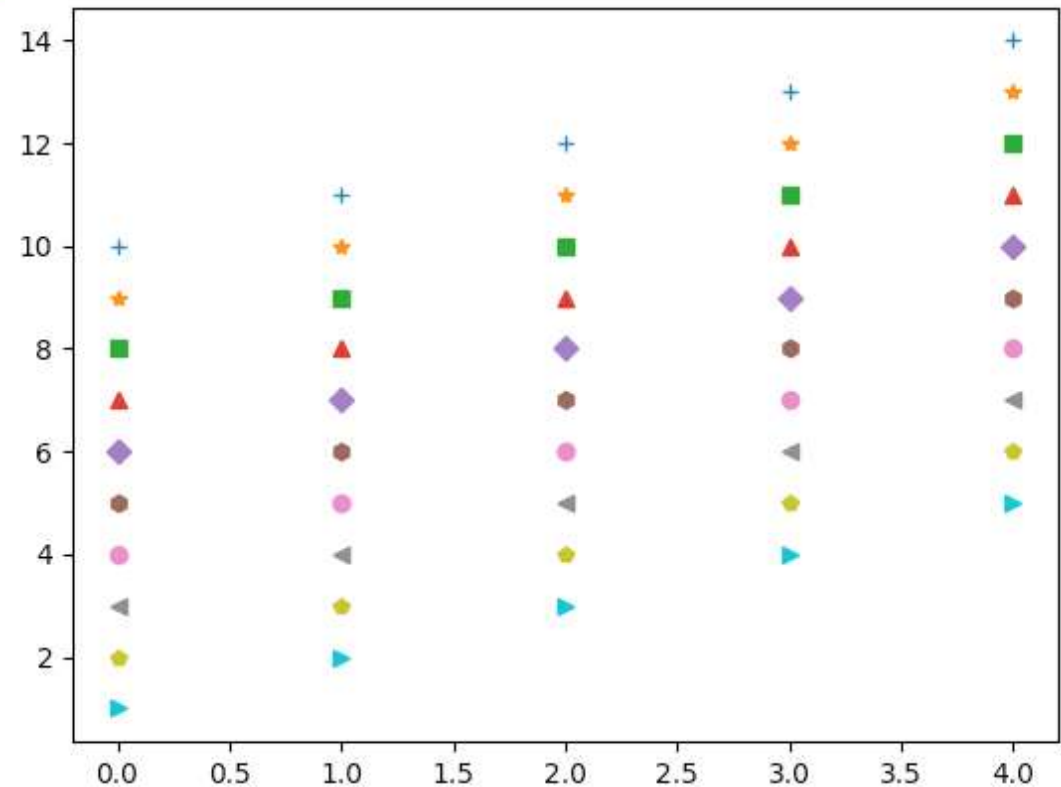
- Metod plot iscrtava oblik na osnovu unete sekvence
- Na iscrtavanje se može uticati različitim parametrima metode

```
plt.plot(  
    [1,2,3,2,4],  
    "o",  
    color='r',  
    linestyle='--')  
plt.show()
```



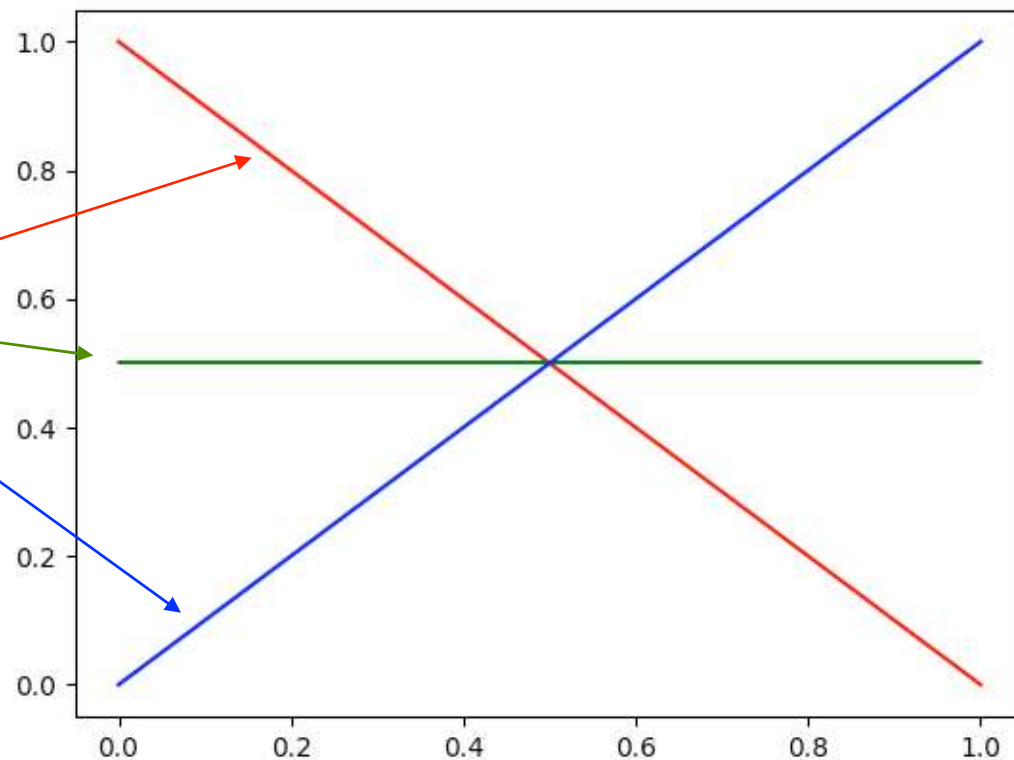
Formatiranje markera

```
plt.plot(
    [10,11,12,13,14], "+",
    [9,10,11,12,13], "*",
    [8,9,10,11,12], "s",
    [7,8,9,10,11], "^",
    [6,7,8,9,10], "D",
    [5,6,7,8,9], "h",
    [4,5,6,7,8], "o",
    [3,4,5,6,7], "<",
    [2,3,4,5,6], "p",
    [1,2,3,4,5], ">"
)
```



Bojanje oblika

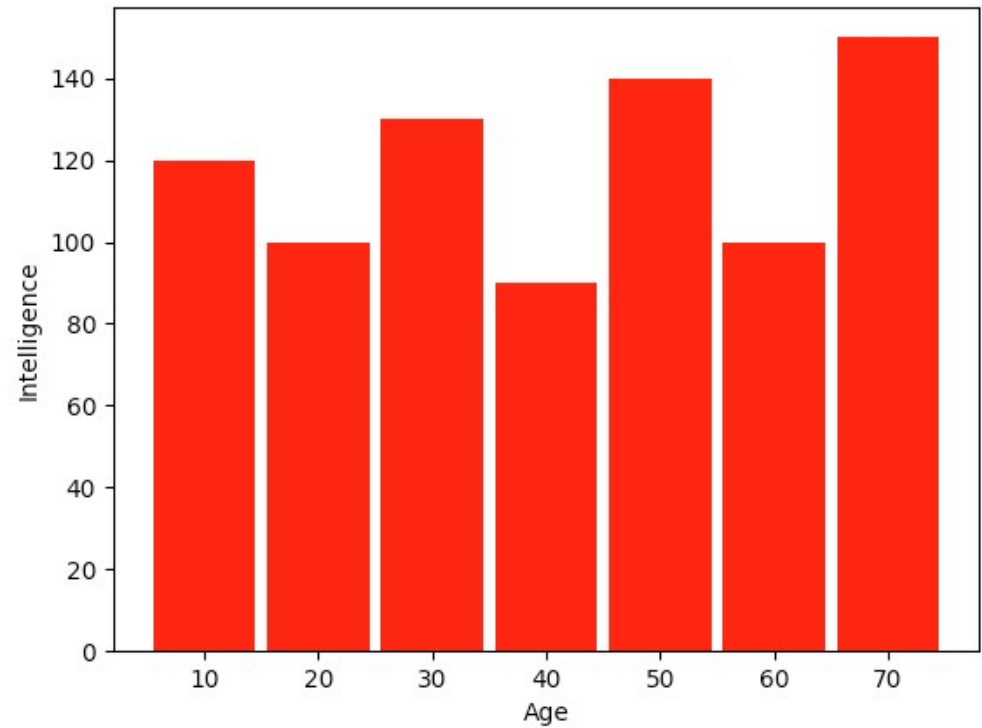
```
plt.plot(  
    [1,0], "r",  
    [0.5,0.5], "g",  
    [0,1], "b"  
)
```



k	black
b	blue
c	cyan
g	green
m	magenta
r	red
w	white

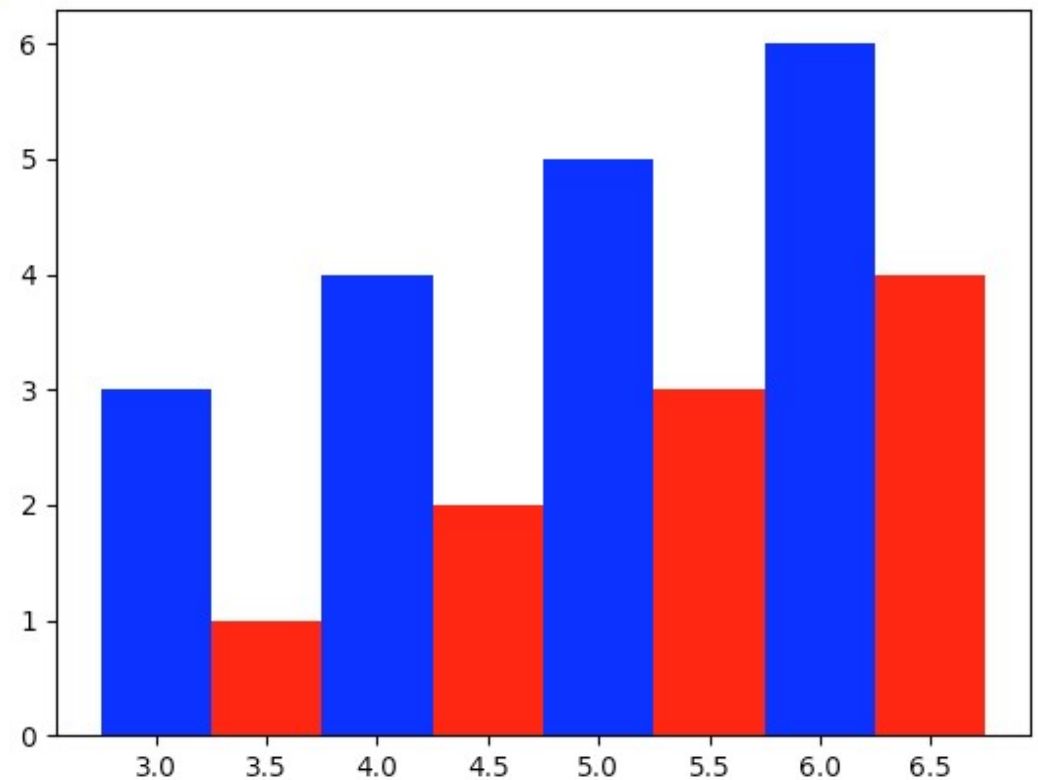
Trake (barovi)

```
a = [10,20,30,40,50,60,70]
b = [120,100,130,90,140,100,150]
plt.bar(
    a,
    b,
    color="red",
    width=9
)
```



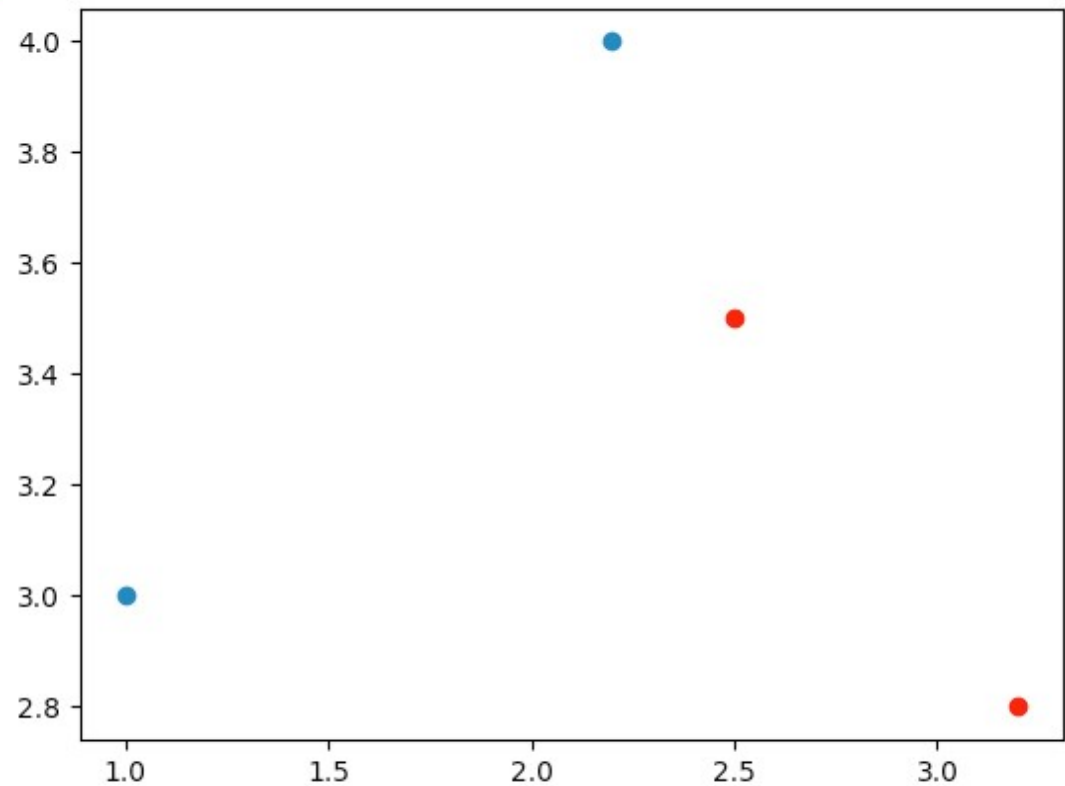
Trake (barovi)

```
plt.bar(  
    [3,4,5,6], [3,4,5,6],  
    color="blue",  
    width=0.5)  
plt.bar(  
    [3.5,4.5,5.5,6.5], [1,2,3,4],  
    color="red",  
    width=0.5)
```



Tačky (scatter)

```
plt.scatter(  
    [3.2,2.5],  
    [2.8,3.5],  
    color="red"  
)  
plt.scatter(  
    [1.0,2.2],  
    [3.0,4.0]  
)
```



Eksport u fajl ili base64 string

base64 eksport

```
import io
import base64
```

```
buff = io.BytesIO()
plt.savefig(buff)
print(buff.seek(0))
print(base64.b64encode(buff.read()).decode("utf-8"))
```

eksport u fajl

```
plt.plot([1,2,3,4])
plt.savefig("myfile.png")
```